REMARKS

Claims 13, 17, 19, 23, 26, 29, 31, 33, and 34 are pending. Claims 13, 17, 19, 23, 26, 29, 31, 33, and 34 are rejected under 35 USC 103(a) as being unpatentable over Burgess (US 5,805,896), in view of Sakurai et al. (US 6,334,076), Kroeger (US 2002/0165723), and Elmqvist ("A Uniform Architecture for Distributed Automation", Advances in Instrumentation and Control, Instrument Society of America, Research Triangle Park, NC US, Vol. 46, Part 2, 1991, Pages 1599-1608).

The independent claims 13, 26, and 33 are amended herein to clarify the claimed limitations. No new matter has been added. Claims 13, 17, 19, 23, 26, 29, 31, 33, and 34 are presented for examination.

Applicants' paragraph numbers mentioned herein are relative to the substitute specification.

Claim amendments

The independent claims 13, 26, and 33 are clarified by reciting that the know-how previously entered into the description of the components includes the predecessor/successor relationships. No new matter has been added. For example, claim 13 already recites "wherein the input/output information comprising predecessor/successor relationships among the components is included in the description". The amendments simply clarify that the predecessor/successor information is part of the know-how previously entered in the description, as supported in paragraphs 22-25 of the substitute specification.

Response to rejections under 35 USC 103(a)

I. The independent claims 13, 26, and 33 recite the generation of plant automation code by graphically mapping input/output information to ports on components based on information, including predecessor/successor relationships, contained in a description of the components. Burgess does not include predecessor/successor information in his component objects. Instead his directed relationships are defined at the graphical mapping stage by the programmer.

Examiner asserts on page 6 of the Office Action of 04-08-2008 that directed relationships of components are defined in Burgess col. 3, lines 29-34, lines 54-57 (FIG 4), and col. 4, lines 1-

16 (FIG 5). However, these lines describe visual programming as illustrated in FIG 4, in which a programmer graphically configures the directed relationships. Since predecessor/successor relationships are not contained in the descriptions of the objects 410, 420, 450, and 460, nothing prevents a reversal of the C-to-F and F-to-C calculator objects by the visual programmer, which would produce incorrect relationships.

> Burgess col. 3, lines 21-38: FIG. 4 is a diagram illustrating visual programming of the present invention. To generate a visual program to implement a temperature converter, a programmer would position a Fahrenheit scroll bar 401, a Fahrenheit display 430, a Centigrade scroll bar 420, and a Centigrade display 440. The programmer would also position an FtoC calculator 460, which converts a Fahrenheit value to a Centigrade value, and a CtoF calculator 450, which converts a Centigrade value to a Fahrenheit value. In one embodiment, the components are selected from an extendible list of available components. The programmer then connects the components through their ports. The connections 412->461 and 412->431 indicate that when the Fahrenheit scroll bar is changed (e.g., slider moved), the new value is sent to the FtoC calculator and the Fahrenheit display. The connection 462->421 indicates that when the FtoC calculator calculates a new Centigrade value, the new value is sent to the Centigrade scroll bar.

In contrast, Applicants' directed relationships are already contained in a description of each component, constraining the connections to a proper order by allowing fewer degrees of freedom in order to reduce the possibility of error.

> Applicants' paragraph 10, lines 1-3: "In the system according to the invention data continuity is achieved in that control-relevant information is already contained in a description."

> Applicants' paragraph 20, lines 13-18: "The information already contained in the description 1 is used to allocate input and output information to the ports. The predecessor-successor relationships between the components are governed by this information, i.e. who sends data to whom via which data input is defined thereby."

This prevents incorrect relationships that are possible in Burgess, because the predecessor-successor relationships are defined prior to graphical mapping stage for local plant code generation.

> Applicants' paragraph 21: "On the basis of the metainformation, the components 2 are connected to one another by automated means. Particular connections between

> the components 2 can only be implemented if this is permitted by the constraints described in the metainformation. Automated "wiring" of the components 2, and therefore automatic generation of automation code, are therefore effected."

Applicants reduce freedom in order to reduce complexity in automation programming, because incorrect options are eliminated from consideration. Continuity of expert information and earlier know-how guides and limits the plant automation code developer.

> Applicants' paragraph 22, all lines: "The work of the development engineer is greatly facilitated thereby, since fewer degrees of freedom exist as a result of the definition of the metainformation, reducing the possibilities of error. In addition, a continuous information flow is ensured, reducing the loss of already established know-how during the development of the automation system."
>
> Applicants' paragraph 34, lines 2-3: automation code is generated on the basis of existing descriptions 1 of a plant structure.

II. On top of page 7 of the office action, Examiner concedes that Burgess does not teach (1) code generation for manufacturing or processing plants and automation code generated on the basis of a structure of the plant and know-how previously input into the description. Know-how previously input into the description is recited in all of the independent claims. It includes predecessor/successor relationships, as taught in paragraphs 22-25 of the specification. This is not simply an intended use as Examiner asserts, but is a feature that prevents sequence errors in any use.

Sakurai provides standard program modules previously coded by program engineers. These standard modules are independent of plant type (col. 4, lines 3-6). A plant operator inputs a plant operation procedure by keyboard, which includes a time sequential control flow (col. 3, lines 61-63). This results in a selection of standard program source code modules for assembly into a load module for execution in a programmable logic controller (PLC). There is no indication that the standard program modules include a previously entered description of predecessor/successor relationship for plant components.

III. Examiner concedes on top of page 7 that (2) Burgess does not teach components described in a drawing comprising control-relevant information in a manufacturing and/or

processing plant. Examiner then cites Sakurai col. 2, lines 20-51. However, these lines do not mention a drawing or graphics at all. Examiner also cites Sakurai col. 4, lines 8-22. However, these lines describe visually monitoring of a PLC program while it is executing, in order to verify proper program operation -- not graphically configuring a specification for automatic code generation.

IV. Examiner concedes on top of page 7 that (3) Burgess does not teach control information described in a drawing based on a material flow in a manufacturing and/or processing plant. Examiner then asserts that Elmqvist supplies the claimed feature of drawings with control-relevant information based on material flow in a processing plant, because it is inherent that the physical objects of the plant form the path for the material or fluid flow as shown in the example of the tank system (figures 1-5). However, as in Burgess, this tank system layout only exists after the visual designer has selected the graphic components and placed them in this order. These graphical components do not have predecessor/successor descriptions stored in them to require an order based on a material flow and prevent mistakes. Instead, the design module definitions of Elmqvist are purely hierarchical (FIG 2). The first line under VISUALIZATION on page 1605 states: "The complete picture, as seen in a window, is a hierarchical picture according to the module hierarchy." The two tanks of FIG 1 are just instantiations of the same "tank" object. Thus their order in FIG 1 could be reversed -- the same kind of mistake discussed for Burger above. It so happens that this reversal would not make a difference in FIG 1 of Elmqvist, but only because FIG 1 is too simple an example to illustrate an ordered relationship. The examples of Elmqvist and Burgess are both highly simplified, but at least the example of Burgess can be used to show how order is important, which is the case in a manufacturing or processing plant.

V. Examiner concedes on top of page 7 that (4) Burgess does not teach system components defined to have predecessor/successor relationships. Examiner then cites par. 112 of Kroeger. This paragraph describes a display of project management task records. However, these tasks may be edited/added at any time, thus changing their sequence (par. 113).

> Kroeger par. 113: As set forth hereinabove, tasks may be edited/added at any time by authorized contacts. Further, the project task manager may provide an audit trail of all task changes made, when they were made, and by whom. The project task manager may also display bar graphs and calendars of schedule tasks & relationships over a scrolling time period. As such, the project task manager may display real time projections of budgets and actual costs.

Kroeger teaches a project and document management system -- not a system for generating plant automation code or any kind of automation code. Examiner has not identified any elements in Kroeger that could correspond to manufacturing plant components. Kroeger simply coordinates tasks done by humans, which does not apply to the present invention. Nor does he provide a graphical configuration process. FIGs 7 and 8 show a text menu-driven screen. Kroeger provides full flexibility for changes in the schedule and contracts (the term "change order" is found 11 times in Kroeger). Flexibility is a primary purpose of Kroeger in order to overcome prior art inflexibility (par. 5, lines 1-5, and par. 6, lines 5-9). As such, this feature is not simply an alternate embodiment. However, it is incompatible with the present invention, which reduces flexibility to avoid mistakes in plant automation code. This teaches away from the present invention, in which the predecessor/successor relationships are pre-established.

Conclusion

The proposed combination of Burger, Sakurai, Kroeger, and Elmqvist does not teach or suggest certain features of the present invention claimed in the independent claims, as argued above. These features provide major benefits in safety and continuity of plant automation design and code generation over the prior art. Furthermore, Kroeger is in a different field, and teaches away from the present invention, as argued above. Therefore, Applicant respectfully requests withdrawal of the 35 USC 103 rejections, and allowance of the present application.

The commissioner is hereby authorized to charge any appropriate fees due in connection with this paper, including the fees specified in 37 C.F.R. §§ 1.16 (c), 1.17(a)(1) and 1.20(d), or credit any overpayments to Deposit Account No. 19-2179.

Respectfully submitted,

Dated: 6/4/08

By: _____

John P. Musone
Registration No. 44,961
(407) 736-6449

Siemens Corporation
Intellectual Property Department
170 Wood Avenue South
Iselin, New Jersey 08830